Automatic numerical data extraction from scientific line- and scatter-plots images

Gabriel L. Cuendet, Peter Staar, Costas Bekas, Maria Gabrani
IBM Research, Zurich
Saumerstrasse 4
8803 Rueschlikon, Zurich
Email: gcu,taa,bek,mga@zurich.ibm.com

***Abstract*—Extracting quantitative information, on the form of numerical data, directly from a graphical representation such as a graph or chart provides access to knowledge otherwise inaccessible in a document. In order to scale when data need to be extracted from millions of documents, a fully-automatic method is needed. We propose such a method that first analyses and understands the structure of the chart and extracts the data in a subsequent step. The understanding of the structure of the chart allows to better handle the important appearance variability of the chart. After detecting structural primitives, we use a rule-based method and a method based on Markov Logic Network (MLN) to semantically label them. The numerical data are then extracted and scaled from image coordinates to their original coordinates. On a test set of 200 synthetic chart images, we show that the rule-based method detects the axes in 100% of the images, with no false positive (FP) whereas the MLN-based method reaches 86.3%, with no FP. All markers are retrieved in 95.1% of the images (with one FP in 23.5% and more than one FP in 6.6% of those) with a relative error between 0.54% and 4.1% of the axis range. Additional qualitative results are presented on real images from the european patent office. Our method to extract numerical data from line- and scatter-plots shows promising initial quantitative results on synthetically generated images and qualitative results on charts images from patents. To the best of our knowledge, this represents the first completely automatic method that extracts original numerical data from charts images.**

I. Introduction

Graphs and charts, such as line- and scatter- plots are commonly used in academic, scientific, financial, patent and other analytical documents in order to graphically represent quantitative information. As examples, the results of scientific experiments or performance of businesses are often summarized using such representations. The underlying numerical data are generally not provided in other data formats in these documents. Extracting that information (*i.e.* the numerical data) directly from the graphical data representation (*i.e.* the graph or chart) is thus of crucial importance and provides access to knowledge otherwise inaccessible.

The vast majority of these graphical data representations are only available as bitmap images. As the graphical symbols (including lines, markers and text) used in graphs and charts are rastered, specific image analysis techniques may be applied in order to first identify these symbols and then extract their semantic and the information they represent. In order to run statistical analysis, identify trends, forecast future behaviors, simulate models or compare their own experiments' results with the data published in a document, researchers, scientists, financial analysts and other users need access to these data in a form that allows a computer to process them. Currently, these users use semi-automatic tools that provide an interactive interface to extract data from figures in digital documents [1]. Semi-automatic methods do not scale when data need to be extracted from millions of documents in large digital libraries.

II. Related work

The applications of charts images analysis are numerous and varied and include information extraction from figures in digital libraries [2], automated vizualization redesign [3], [4] document retrieval and chart images retrieval [5], biomedical knowledge discovery [6], textual summary generation for visually impaired people [7] and quality assessment of data charts [8]. In these applications, the three main images source domains considered are patent documents from the patent offices, as in [9], scholarly works that are scanned or manually collected online [10] and web documents [11].

Given a document, two main steps are needed in order to extract information from the charts images it contains. First, images are detected and classified into predefined classes [9], [10]. Second, the data are extracted, taking into account how data are represented within that class. As this work focuses only on that second part, we will restrict our related work review to the topic of data extraction from charts.

Molla et al. [12] first described a simple system to extract data from a line chart. Their system suffers severe limitations: it is limited to a single line per chart, it is unable to handle even slightly rotated charts due to the axes detection method based on histograms of black pixels along lines and columns and the numerical data do not correspond to the original data but are expressed in the image coordinate system, in pixels.

An important work towards automatic extraction of data from 2D line- and scatter-plots is presented in [13], [10]. The authors are the first to propose a method to extract multiple data series from a chart (several lines in a line-plot and different markers in a scatter-plot). No quantitative evaluation of their method is described. Instead, they visually assess the similarity of the lines between the input image and an image generated from the extracted data. Similarly for scatter-plots, they visually verify that the number of extracted markers

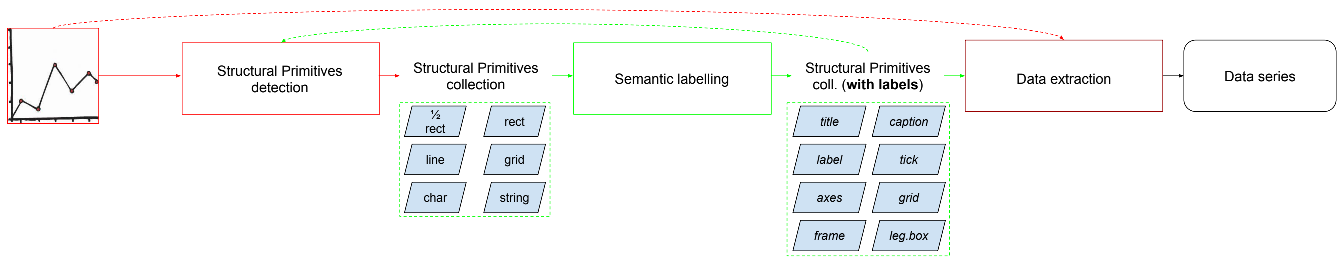


Fig. 1. Complete pipeline of the proposed method. *In red* are the steps taking pixel level representation as input whereas *in green*, the input is at the *structural primitives* level.

corresponds between these two images and that their shapes are preserved. Again, in both cases, actual numerical data are not extracted due to the scale of the axes not being detected.

Savva et al. proposed an automated system to redesign bar and pie charts in [3]. In order to redesign a given chart, or even represent the same data in a different type of chart, they accurately extract the structure of the chart for 71.4% of the charts and successfully recover numerical data for 67.1% of those. Following the idea of redesigning charts, Baucom et al. [4] proposed *ScatterScanner*, a system that allows to interactively redesign scatter-plots. Their system is limited to clean scatter-plots, without gridlines or text annotations, containing only one data serie represented with simple shape markers and plotted in the first quadrant of the Cartesian plane. For 62.5% of the scatter-plots respecting these assumptions, they reported that over 95% of discernable points were correctly identified with no false positives.

Very recently, Choudhury et al. described an architecture for information extraction from figures in digital libraries [2]. Their work includes a module for semi-automatic numerical data extraction from figures. The data extraction is thus not automatic but the user needs to indicate the beginning and ending points of X and Y axes (by recording mouse clicks) and axis scales (linear/logarithmic). Then, curves plotted in different colors are extracted, but binary or grayscale curves pose greater challenges and require user's input. The authors clearly state that up-to-date no complete algorithm exists for automatic reliable and scalable data extraction.

In [14] the authors present a complete pipeline (including the binary classification of graphic images into line-plots vs. the rest) to extract components of a line plot *i.e.* axes, legend and labels and the individual lines represented. They consider solid, dashed and colored curves and do not have any restrictions on the number of curves but their method is not able to handle plots with grids. Again, the data are not directly extracted as the axes are not scaled. Their evaluation is limited to comparing the shapes of the original curves and of the reconstructed curves. They report a classification accuracy of 91% and an axes extraction recall of 92%.

In summary, the first major limitation of existing methods is their inability to consider real charts images, possibly with a grid or a legend in the data area, or text annotations. Existing methods make very restrictive assumptions regarding the structure of the chart such as the absence of a grid or any other element in the data area that is not data. This is done in order to reduce as much as possible the important appearance variability. A second limitation of previous works lies in the fact that real numerical data, in original data coordinates are never extracted. The data thus need to be converted to real scale data manually.

III. Method

To overcome these limitations, we propose to first analyse and understand the structure of the chart and to extract the data in a subsequent step. The understanding of the structure of the chart provides important information, such as whether a grid is present, or a legend box and allows to handle them correctly when extracting the data.

Fig. 1 shows a complete high-level overview of our method. In order to build a precise and complete understanding of the structure of the plot, we first detect *structural primitives* from the image. These are then labelled according to their *semantic*. As an example, a straight horizontal line could be detected in the first step and would be added to the collection of structural primitives. Depending on its position, size and neighboring primitives, that line could be the *x-axis* of the plot and would be semantically labelled as such. As depicted in Fig. 1, these two steps can be repeated. This allows to detect primitives while benefiting from the already detected and labelled primitives. Finally, after each element in the image has been labelled, data are extracted. The understanding of the structure of the chart allows to extract data from the correct region of the image (*i.e.* from the *data area*, within the axes) and to remove elements in that region that are no *data* such as a legend box and its content or a grid. Thus, we don't need to make the strong assumption of a clean data area and are able to better handle the important appearance variability due to the structure of the chart.

A. *Structural primitives detection*

The constituent primitives of the image are detected in two steps: first, characters and straight lines are detected, and second, some of these elements are grouped together based on their topology to form new, more complex elements such as strings of text (from characters) and rectangular frames or grids (from straight lines).

1) Characters and straight lines detection: For character detection, we use the method described in [15] and implemented in the OpenCV 3 library (opencv-contrib). We detect straight lines using the Hough transform [16]. These lines are parametrized by the radius ρ and angle θ of their normal intersecting the origin. The Hough transform constructs the $\rho - \theta$ parameters space and uses a voting-based technique, which results in a 2D histogram of $\rho - \theta$ parameters.

More specifically, the image is first thinned using the Zhang-Suen algorithm [17]. Then, lines are detected at specific angles by specifying a reduced parameter subspace along the θ dimension of the histogram in the Hough transform. Typically, we only keep values of θ in an interval around the specific angle that we are looking for. The width of that interval depends directly on the desired tolerance for skewness of the image. This not only allows to search for lines at specific angles, but also improve both the memory and computational efficiency of the Hough transform. In order to compute the pixels support of the detected lines, we compute the intersection between each line and the image and assign the pixels in the intersection to the given line. In order to remove distinct lines sharing most of their pixels, lines are then filtered. If two lines share a number of pixels greater than a threshold, only the line with the largest total number of pixels is kept. Finally, disconnected small components of the lines are removed.

2) Grouping of characters and straight lines: Single characters are grouped together into strings using a set of rules that we defined. The challenge when grouping characters comes from the unknown orientation of each character and the relatively high probability of a string being oriented vertically (i.e. rotated by 90 degrees with respect to the orientation of the image) . Moreover, strings with different orientations can be spatially close to each other in the image (for example the y-axis label, usually oriented vertically and the y-axis ticks values, usually oriented horizontally).

Our proposed grouping algorithm proceeds sequentially, by first grouping pairs of characters horizontally when two characters satisfy the following set of rules:

- The height ratio of the smallest character over the largest one must be greater than a threshold corresponding to the typographic x-height over the ascender-height.
- The angle formed by the segment linking each character centroid and a reference axis (horizontal for horizontal text, vertical otherwise) must be within an interval defined by two thresholds. This interval is bounded by the angle formed by two successive characters: one with an ascender and one with a descender.
- The signed horizontal distance between both characters must be within an interval defined by two thresholds.
- The signed vertical distance between both characters must be within an interval defined by two thresholds. These indicate how much successive characters can deviate from the typographic baseline.

The same set of rules is then used to group pairs of characters vertically with the additional condition that neither element of a valid pair should be part of a horizontal grouping of length greater than a threshold. That way, we favor horizontal groupings over vertical groupings. The detected strings are then fed to the Tesseract optical character recognition (OCR) engine and their text is stored. Numerical values are also extracted, in cases where the string only contains numerical values (typically the axes thicks).

In a similar fashion, subsets of straight lines are grouped together into either half-rectangles, rectangles or a grid structure using a set of rules. These rules specify at which angle the lines potentially part of the new primitive should intersect (typically in an interval around $\pi/2$) and which parts of the lines should intersect: their extremities in the case of a rectangle or at periodic distances in the case of a grid.

B. Semantic labelling

Once structural primitives have been detected, the understanding of the image structure is provided by the semantic labelling of those primitives. This basically consists of mapping structural primitives to semantically meaningful objects. This mapping allows to answer questions about the image structure such as which text string is the title of the figure, or the x-axis label, or which rectangle defines the axes or the legend box or the image frame, etc.

One such mapping can be defined by rules. Rules-based labelling is particularly simple but fails to address the large variability in plots' structure. Moreover, the semantic labelling of structural primitives is a *collective classification* task [18]: the labels of different primitives are not independent. A rules-based classification system would need complex feedback loops in order to label primitives according to other primitives' labels.

Probabilistic graphical models enable us to efficiently handle uncertainty and statistical dependencies between primitives. Given a Markov network (also called Markov random field) modelling the statistical dependencies between a set of variables including observed variables (*i.e.* features that can be extracted from the detected structural primitives) and the unobserved labels, the labelling task becomes a *Maximum A Posteriori* (MAP) problem. The main challenge here is to correctly define the structure of the network as well as the dependencies between variables. We use *Markov Logic Network* to that end.

A Markov Logic Network (MLN) is a set of weighted first-order logic formulas [19], [20] and can be viewed as a template for constructing Markov networks. A first-order domain is defined by a set of constants (that is assumed finite) representing objects in the domain and a set of predicates representing properties of those primitives and relations between them. In our application, the set of constants contains the structural primitives and the labels. Examples of predicates that we defined are `HasLabel(x, AXES)` used to label the structural primitives or `Contains(x, y)` indicating wether the structual primitive x contains y or not.

A predicate can be grounded by replacing its variables with constants. A world is an assignment of a truth value to each possible ground predicates (or atoms). A first-order knowledge base (KB) is a set of formulas in first-order logic, constructed from predicates using logical connectives and quantifiers. In first-order logic, the KB is a set of hard constraints on the set of possible worlds. Whenever a world violates one formula, it has zero probability. The idea in MLN is to soften these constraints and add the ability to handle uncertainty and tolerate imperfect and contradictory knowledge. MLN can be considered as a log-linear model with one node per ground atom and one feature per ground formula. The joint distribution over possible worlds x is given by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{n} w_i f_i(x_{\{i\}})\right) \qquad (1)$$

where Z is the partition function, F is the set of first-order formulas in MLN and n is the number of formulas in F, $x_{\{i\}}$ is a state of ground atoms appearing in the formula F_i and the feature function $f_i(x_{\{i\}}) = 1$ if $F_i(x_{\{i\}})$ is true and 0 otherwise. The weights w indicate the likelihood of the formula being true. A world that violates one formula in the knowledge base is less probable than one that does not violate any formula but not impossible.

For instance, the knowledge "A rectangle that contains another rectangle labelled as *axes* is the *image frame*" can be described using the formula $\forall r_1, r_2, Contains(r_1, r_2) \wedge HasLabel(r_2, \text{AXES}) \Rightarrow HasLabel(r_1, \text{IMAGE\_FRAME})$. We define a KB with such formulas, using our understanding of what the structure of a plot is. At training time, the weight of each formula in the KB is learned from a training set of artificially generated images. At labelling time, MAP inference is performed on the Markov network and the most probable label is assigned to each structural primitive.

C. Numerical data extraction

Once the structure of the chart is known, data are extracted from the data area. If present, the grid, the legend and text are first removed from inside of the axes. The resulting image is rotated in order to compensate for any detected skewness given by the angle at which the axes rectangle is oriented. Finally, only the inside of the axes is kept by cropping the image. The resulting image ideally only contains data: either markers, in the case of scatter plots, or lines with markers superimposed, in the case of line plots. In this case, the markers often represent experimental evidence, such as measurement points, and the lines depict the inferred model. In both cases the markers are of main interest.

From the cropped image of the inside of the axes we use mathematical morphology in order to remove the lines and keep only the markers in the image. We perform an opening operation with a round structural element whose size is slightly larger than the width of the lines to be removed. Finding the center of each connected component in the data area finally gives us the coordinates (in image coordinates) of the data points corresponding to the markers.

Finally, we convert the obtained data from image coordinates (in units pixels) to data coordinates (in whichever units the data are). Given the positions of the centre of the ticks and the corresponding numerical value of each of them, the conversion of coordinates is a 1 dimensional linear scaling for linear axes and a log scaling for log axes. In order to be robust to eventual errors in the OCR, resulting in a wrong numerical value for the tick, we use the *random sample consensus* (RANSAC) algorithm [21] to compute the scaling. Compared to the method described in [4], the advantage of our method is that we benefit from all detected ticks to make the scaling more robust to outliers. We obviously need at least two of them for each axis but these two can be any two ticks and not necessary the top, bottom, leftmost and rightmost values.

IV. Experiments and Results

In order to demonstrate the validity of our method, we need a set of charts images with corresponding ground truth labels for each of them. By ground truth labels, we mean the numerical values of the data that are plotted, but also information about each structural primitive in each chart, such as its type, position, size and label or the string that a text primitive contains. To the best of our knowledge, there exists no such publicly available database. Annotating figures manually is a very tedious task and the accuracy of the annotations (especially for the numerical values of the data) largely depends on the quality of the image. For these reasons, we use synthetic data, *i.e.* charts images generated with the python plotting library Matplotlib [22].

A. Results on synthetic data

In order to mimic the large variability of line- and scatter-plots appearances, we parametrize the structure of the plot and for each new figure that we generate, we randomly draw its parameters from each parameter's distribution. The legend text, the axes labels and the title are randomly generated, with the number of words sampled from a uniform distribution in an interval $[n_{min}, n_{max}]$ and each word length sample from a uniform distribution in a different interval. Examples of other random parameters include: the likelihood of the figure having a title, the likelihoods of the location of that title$["right", "centre", "left"]$ and the likelihoods of the font-sizes $["medium", "large", "x - large", "xx - large"]$, the likelihoods of the markers being one of the 14 different shapes whose sizes are sampled from a uniform distribution on the interval $[size_{min}, size_{max}]$, the likelihoods of each axes having a grid and a given grid style $(["solid", "dashed", "dash - dot", "dotted"])$, the likelihood of the figure having a frame, the likelihood of the figure having a legend and the likelihoods of that legend being in each one of 16 different locations (5 outside the axes and 11 inside the axes). The range of the y-axis is randomized whereas the range of the x-axis is between 0 and 100.

We generate a test set of 200 images, with the parameters of each of them randomly sampled from their distributions. We then apply our method on each image in order to retrieve its structure and the coordinates of each data point. The retrieved structure and data points are then compared to the

		precision	recall
Rule-based method	title detection	1.0	0.986
	axes detection	1.0	1.0
MLN-based method	title detection	0.304	0.93
	axes detection	1.0	0.863

TABLE I
RESULTS ON 200 IMAGES FOR THE STRUCTURE ANALYSIS

	x-axis	y-axis
Rule-based method	0.54%	4.10%
MLN-based method	1.00%	2.80%

TABLE II
RESULTS ON 200 IMAGES FOR THE DATA EXTRACTION ACCURACY

ground truth. Two variants of the proposed method are tested: the rule-based method uses hard rules to label the structural primitives whereas the MLN-based method uses the described Markov Logic Netwok framework. We measure and report the precision and recall for the axes detection as they are the most important structural primitive as well as for the title detection. These results are reported in table I. To evaluate the data extraction accuracy, we compute the percentage of images for which all markers are retrieved and the accuracy of the data extraction. The accuracy of the data extraction is computed in the following way: the absolute error (in data coordinates) between the extracted data points and the ground truth data points is computed independently along each axis. These errors are then normalized by the range of the corresponding axis and averaged over all data points. Table II thus presents the mean relative errors along each axis for both labelling methods' runs.

All markers are retrieved in 95.1% of the images. In 69.9% of those images, there is no false positive, in 23.5% there is one and in 6.6% there are more than one false positives. The relatively high number of images with one false positive is due to the fact that the legend box might not be detected and removed from the data area properly. As our test images contains only one data serie, the legend box is small and thus more difficult to detect. It contains only one marker which is the false positive. Amongst the 4.9% of images in which we miss some markers, we miss only one in 85.7% of them. This is due to part of the marker being invisible (when located at the boundary of the plot) or to the fact that the legend can be placed on top of a marker as its position is random.

B. Results on CLEF-IP 2011 challenge dataset

In addition, we also tested our method on figures from the publicly available training set of the graph category of the CLEF-IP 2011 challenge dataset[1] [23]. This dataset does not provide any ground truth, neither for the structure nor for the data points. We thus only assess visual similarity of the data between the original chart and a chart that we regenerate using the extracted data points. Figure 2 presents our results on two chart images from that dataset, one line-plot and one scatter-plot. In the first example, the line-plot from EP0731067, all markers are correctly retrieved and both axes are correctly scaled. In the second example, the scatter-plot from EP1313794, the grid is properly removed, all markers are correctly retrieved but the y-axis is not scaled properly, due to the OCR that does not recognize the ticks values correctly.

Even though it is difficult to perform a quantitative assessment on these data, these two examples show encouraging qualitative results and demonstrate that our method can be applied on real images.

V. CONCLUSIONS AND FUTURE WORK

We present a methodology to extract numerical data from line- and scatter-plots and show promising initial quantitative results on synthetically generated images and qualitative results on charts images from patents.

Our structural analysis (structural primitives detection and their semantic labelling) allows us to handle real charts images, possibly with a legend, grid or text annotations. We also propose a complete method to retrieve the scale of the axes and extract data in their original data coordinates and take advantage of RANSAC to handle outliers in the detected ticks and their value. To the best of our knowledge, this represents the first completely automatic method that extracts original numerical data from charts images.

The rules-based method seems to work well on our synthetic data benchmark but exhibits strong limitations in terms of generalization and does not allow for collective classification. In order to overcome these limitations, we propose to use the Markov Logic framework to generate a Markov network from a set of first-order formulas and training examples. More work is required on the formulas definition, as the implications of a given set of first-order formulas on the probability distribution modeled by the Markov network is not intuitive, as discussed in [24]. The generation of a more representative training set would also contribute to the improvement of the performance of the system. When there are more than one data series represented, future work will include the use of clustering techniques on the markers' shapes in order to identify to which data series each marker belongs to.

Finally, we believe that the described framework exhibits a great potential for generalization and want to extent this work to other types of charts images such as pie charts or bar plots.

[1]available at: http://www.ifs.tuwien.ac.at/c̃lef-ip/download/2011/data/img-cls_training/gr .tar.bz2

REFERENCES

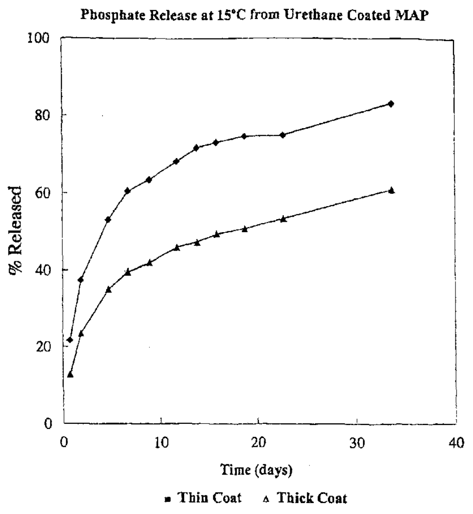
[1] S. Rakap, S. Rakap, D. Evran, and O. Cig, "Comparative evaluation of the reliability and validity of three data extraction programs: UnGraph, GraphClick, and DigitizeIt," *Comput. Human Behav.*, vol. 55, pp. 159–166, feb 2016.

[2] S. R. Choudhury and C. L. Giles, "An Architecture for Information Extraction from Figures in Digital Libraries," in *WWW '15 Companion Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 667–672.

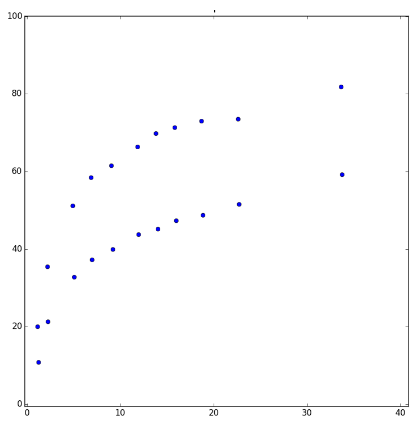
[3] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "ReVision: Automated Classification, Analysis and Redesign of Chart Images," in *Proc. 24th Annu. ACM Symp. User interface Softw. Technol. (UIST '11).* New York, New York, USA: ACM Press, 2011, pp. 393–402.

[4] A. Baucom and C. Echanique, "ScatterScanner: Data Extraction and Chart Restyling of Scatterplots," in *CHI'13 Conf. Hum. Factors Comput. Syst.*, 2013, pp. 1–8.

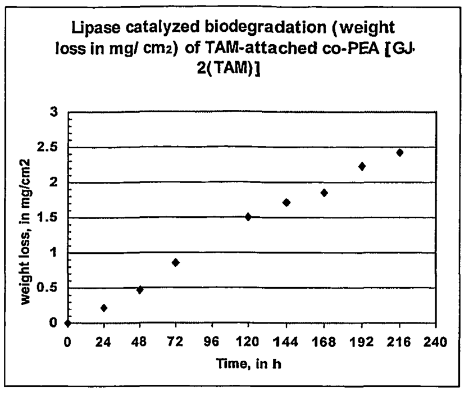
[5] M. Lupu, "Patent Retrieval," *Found. Trends Inf. Retr.*, vol. 7, no. 1, pp. 1–97, 2013.



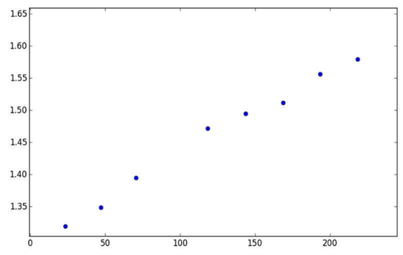
(a) Original image from EP0731067



(b) Reconstructed image for EP0731067



(c) Original image from EP1313794



(d) Reconstructed image for EP1313794

Fig. 2. Examples from the CLEF-IP 2011 challenge dataset

[6] B. Cheng, R. J. Stanley, S. Antani, and G. R. Thoma, "Graphical Figure Classification Using Data Fusion for Integrating Text and Image Features," in *2013 12th Int. Conf. Doc. Anal. Recognit.* IEEE, aug 2013, pp. 693–697.

[7] P. Moraes, G. Sina, K. McCoy, and S. Carberry, "Evaluating the accessibility of line graphs through textual summaries for visually impaired users," in *Proc. 16th Int. ACM SIGACCESS Conf. Comput. Access. - ASSETS '14.* New York, New York, USA: ACM Press, 2014, pp. 83–90.

[8] S. Shukla and A. Samal, "Recognition and quality assessment of data charts in mixed-mode documents," *Int. J. Doc. Anal. Recognit.*, vol. 11, no. 3, pp. 111–126, dec 2008.

[9] N. Bhatti and A. Hanbury, "Image search in patents: a review," *Int. J. Doc. Anal. Recognit.*, vol. 16, no. 4, pp. 309–329, dec 2013.

[10] X. Lu, S. Kataria, W. J. Brouwer, J. Z. Wang, P. Mitra, and C. L. Giles, "Automated analysis of images in documents for intelligent document search," *Int. J. Doc. Anal. Recognit.*, vol. 12, no. 2, pp. 65–81, jul 2009.

[11] J. Gao, Y. Zhou, and K. E. Barner, "View: Visual Information Extraction Widget for improving chart images accessibility," in *2012 19th IEEE Int. Conf. Image Process.* IEEE, sep 2012, pp. 2865–2868.

[12] M. K. I. Molla, K. H. Talukder, and M. A. Hossain, "Line Chart Recognition and Data Extraction Technique," in *Lect. notes Comput. Sci.* Berlin Heidelberg: Springer-Verlag, 2003, pp. 865–870.

[13] X. Lu, J. Wang, P. Mitra, and C. Giles, "Automatic Extraction of Data from 2-D Plots in Documents," in *Ninth Int. Conf. Doc. Anal. Recognit. (ICDAR 2007)*, vol. 1. IEEE, sep 2007, pp. 188–192.

[14] R. R. Nair, N. Sankaran, I. Nwogu, and V. Govindaraju, "Automated analysis of line plots in documents," in *2015 13th Int. Conf. Doc. Anal. Recognit.*, no. ii. IEEE, aug 2015, pp. 796–800.

[15] L. Neumann and J. Matas, "Real-time scene text localization and recognition," in *Comput. Vis. Pattern Recognition, IEEE Conf.*, 2012, pp. 3538–3545.

[16] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[17] T. Y. Zhang and Y.-S. Chen, "A modified fast parallel algorithm for thinning digital patterns," *Comm. ACM*, vol. 27, no. 3, pp. 236–239, feb 1988.

[18] B. Taskar, P. Abbeel, and D. Koller, "Discriminative Probabilistic Models for Relational Data," in *Proc. Eighteenth Conf. Uncertain. Artif. Intell.*, dec 2002, pp. 1–18.

[19] P. Domingos, S. Kok, H. Poon, M. Richardson, and P. Singla, "Unifying Logical and Statistical AI," *AAAI*, vol. 6, no. 1, pp. 2–7, oct 2006.

[20] M. Richardson and P. Domingos, "Markov logic networks," *Mach. Learn.*, vol. 62, no. 1-2, pp. 107–136, 2006.

[21] M. a. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, jun 1981.

[22] J. D. Hunter, "Matplotlib: A 2d graphics environment," *IEEE Comput.Sci. Eng. Mag.*, vol. 9, no. 3, pp. 90–95, 2007.

[23] F. Piroi, M. Lupu, A. Hanbury, and V. Zenz, "CLEF-IP 2011: Retrieval in the Intellectual Property Domain," in *CLEF (noteb. Pap.*, 2011.

[24] D. Jain, "Knowledge Engineering with Markov Logic Networks: A Review," *DKB 2011 Proc. Third Work. Dyn. Knowl. Belief*, 2011.